23 Software Fundamentals

HALIT EREN

PARTIAL LIST OF SUPPLIERS

Automation Onspec Software (http://www.automationonspec.com)
Biosoft Software for Science (http://www.biosoft.com)
BSOFT Software (http://www.bsof.com)
Chempute Software (http://www.compusense.com/)
Creon Software (http://www.creon.com)
Curvaceous Software Limited (http://www.curvaceous.com)
Engineered Software (http://www.eng-software.com)
ExperTune PID Analysis and Tuning Software (http://www.expertune.com)
Industrial Automation Technology (http://www.askiat.com)
Intelligent Optimization Group (http://www.gefanucautomation.com/)
Madison Technical Software (http://www.madisonsoftware.com/)
MTD Software (http://www.mdtsoft.com/)
National Instruments (http://www.ni.com/)
OSI Software (http://www.osisoft.com)
Parijat Cotrolware (http://www.parijat.com)
ProcessPro Manufacturing Software (http://www.processpromfg.com)
Rockwell Software (http://www.rockwellautomation.com/rockwellsoftware)
Software Toolbox (http://www.softwaretoolbox.com)
Visual data Interface (http://www.vdisoft.com/nn/)

INTRODUCTION

There are many definitions of software. The definition preferred by the author is “A generic term for computer programs, which includes systems programs that operate the computer itself, and applications programs that control the particular task at hand.” Software is a set of program instructions that make the computer do some task, such as word processing, manage databases, control plants, etc. It is not only the codes that a programming language generate but it also includes requirements, specifications, designs, program listings, procedures rules, measurements, and data. It also includes tools used to create, test, optimize, and implement the software.

The effective and appropriate use of software is very important in process and automation industries. Software Engineering Institute (SEI) indicates that software development failures have brought down entire companies, such as the $5 billion drug distribution firm in Texas, which declared bankruptcy as a result of a poorly implemented resource-planning system. Organizations and governments worldwide spend about $1 trillion a year on IT projects. Recent data suggested only about 35% of those projects are likely to be completed on time and on budget, with all their originally specified features and functions. Many projects, perhaps 20%, will be abandoned, often after multimillion-dollar investments and the biggest projects tend to fail most often (http://www.sei.cmu.edu/solutions/softwaredev/).

Process control and automation require more than one program and diverse range of software packages. One or more programs that run in tandem to solve a common problem collectively form a *system*.

Software has its own niche industry made up of different entities and people that produce software, and as a result there are many software companies and programmers in the world specialized in certain areas. There are also many nonprofit software organizations like the Free Software Foundation and Mozilla Foundation. Some of the well-known software companies include Microsoft, Novell, Oracle, Symantec, Adobe Systems, and Corel. Process and Automation industry primarily uses software produced by other companies such as PLC manufacturers, SCADA producers, etc.

Software development and implementation constitute the heart of the operations of modern processes and automation. The subject is very broad. However, in this chapter, various aspects of software relevant to the process industries are highlighted briefly and the relevant standards are discussed.

PROCESS CONTROL AND AUTOMATION SOFTWARE

Process control and automation require more than one program and diverse range of software packages. One or more programs that run in tandem to solve a common problem collectively form a *system*. Types of software used in plants, processes, and automation can broadly be listed as follows:

- Custom software to address the needs of individual customers
- Process control software

- Statistical process control software
- Manufacturing software
- Industrial automation software
- Resource management software
- PLC, SCADA, HMI software
- Robotics software
- Real-time data collection and analysis software
- Network asset management software
- Data acquisition and logging
- Virtual instrumentation software
- Data analysis and synthesis software
- Simulations and modeling software
- Health and safety management software
- Computer-aided design software
- Computerized sensory software
- Alarm management software
- Enterprise application integration software
- PID analysis and tuning software
- Optimization software
- Artificial intelligence and fuzzy logic
- Intelligent optimization software
- Laboratory instrumentation and automation software
- Embedded systems software
- Operating systems
- Network management software
- Information systems and information handling software
- And, many more

Some of this software will be provided by third-party companies as complete, packages and others may need to be custom developed or developed in-house. Given the diverse range of software used in processes and automation, it is impossible to give details on the individual software. In this chapter, a broad outline on the fundamentals of the software will be discussed. However, more information can be found in the relevant chapters of this book.

COMMON CHARACTERISTIC OF SOFTWARE

Common to most software, they provide the capability to code logical constructs that has the following capabilities:

1. *User interface* is a mechanism that the end user can input, view, manipulate, and query information in a computer system. At the moment, graphical user interface (GUI) is a very popular technique.
2. *Program control* enables comparisons, branching, calling other programs, and iterations.
3. *Message processing* is the way that a program responds to the end user and provides the ability to pass information to other program objects. Error-message, for example, is an automatic capability to notify and then to recover from an error during input, output, calculation, and reporting.
4. *Model calculations* are step-by-step procedures for solving a problem such as process control, input/output handling.
5. *Object* are physical or abstract things that can communicate with each other through a communication medium. When objects are brought to life, they become attributes, which can do things and interact with other objects. For example, in a controls system, an object may contain a function to set the PID controller and at the same time close the valves.
6. *Moving data* is the storage and retrieval of data from the data structure. Data can be moved from an external source to internal data structure or vice-versa.
7. *Data declaration* describes the data and data structures to a program.
8. *Database* is an easily accessible way of storing data in a relational, hierarchical, network, and object-oriented format. Database is easily accessible by the users and it can be updated, displayed, queried, and printed by the users with appropriate accesses.
9. *Tools* are the programs used to design, develop, analyze, test, and maintain the program design. It includes code generators, compilers, editors, database management, GUI builders, debuggers, and operating systems.
10. *Real-time* is a software system that satisfies critical timing requirements by performing tasks within a specific deadline in connection with another process outside the computer.
11. *Distributed system* is a number of independent and interconnected processes that can cooperate. A good example is the client–server model in which a client initiates a distributed activity and a server carries out that activity.
12. *Documentation* describes how each program within a large system operates and can be used. It includes specifications, design, operation of programs, and description of requirements in a written or generated form.
13. *Simulation* is representation of selected characteristics of the behavior of a physical or abstract system by another system. For example, behavior of process controller can be simulated in another program such as the MATLAB®.

SOFTWARE TOOLS

Software available in most of the microcomputers includes the following:

1. *Assemblers* are used to generate machine code (object code) from the assembly language text (source code). The assembler reads the source text twice, which is known as the *two-pass assembler.*

   On the first pass, the assembler generates as symbol table, which is stored in the RAM. This table is used to equate each symbol address (label) with an absolute address. On the second pass, the assembler generates machine code for each instruction. Although various assembler directives can be included within the source code, these pseudo-mnemonics are not translated into object code. They are used to allow the user to modify object codes produced.

2. *Editors* allow the user to create easily modifiable text files. They are available in two forms: simple line editors operating on a single line of text at a time, and full-screen editors that allow full cursor control over the whole screen. *Editors* are used to create text (such as ASCII) for programs written in a variety of high-level languages.
3. *Interpreters* translate programs written in high-level languages, usually in ASCII text file, into machine-executable code. An interpreter acts on each statement of source text at a time and executes it immediately before fetching the next text line of source text. This gives an advantage for discovering flaws and errors in the source text immediately. Statement containing an error will not be executed and an appropriate error message is generated. Flowchart of an interpreter is depicted in Figure 23.1a.
4. *Compilers* are used to generate machine codes from source text written in high-level languages to be loaded in executable forms. Once the program has been fully debugged, the program is saved, loaded, and run in machine code only once, thus making compilers much faster than interpreters. If an error occurs or a change is made to a program, entire compilation process must be repeated. Flowchart of a compiler is depicted in Figure 23.1b.
5. *Linkers and loaders* link a program with the necessary I/O or mathematical routines after the compilation. The necessary routines are then added to the machine code. During compilation and assembly, symbolic labels are used to represent addresses. Loaders replace the symbolic addresses with absolute addresses to be used for final execution.
6. *Debuggers* provide means of testing programs during run time. They have many features, some of which are as follows: display the contents of a given block of memory in hexadecimal or ASCII, commencing the execution with optional break points, disassembling a block memory, reading a disc file into memory, and examining and modifying the central processing unit (CPU) registers.

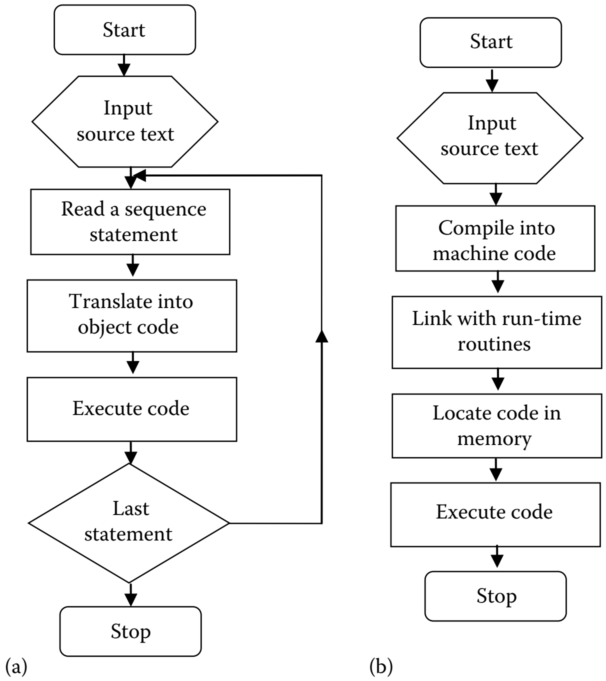

FIG. 23.1
(a) Flowchart of an interpreter and (b) flowchart of compiler.

OPERATION OF SOFTWARE

Software is an essential component of digital systems functionality as they are configured from two main components: the hardware and the software. *Hardware* encompasses the physical interconnections and devices required to store and execute (or run) the software. We can divide computers into seven main streams: (1) personal computers, (2) work stations, (3) mobile computers and mobile devices, (4) servers, (5) mainframes, (6) supercomputers, and (7) embedded computers and embedded controllers.

As explained above, the *software* is a term for describing the role of programs, procedures, and documentation in programmable digital devices and computer systems. They are written and developed using programming languages, scripting languages, microcode, or FPGA configurations. Some examples of programming languages are C or C++, Java, Basic, Fortran, COBOL, etc. The software includes

- *Application software* such as Excel, word processors, video games, Open Office, etc., which provides tasks for users
- *Firmware* is software programmed resident to in programmable memory devices on board mainboards or other types of integrated hardware carriers
- *Middleware* controls and coordinates distributed systems
- *System software* such as operating systems that interfaces with hardware to provide a coordinated service for applications such as Linux, Microsoft Windows, etc.
- *Software testing* consists of various methods to test and declare a software product fit before it can be used
- *Testware* is a package for testing utilities and application software
- *Web pages* developed by technologies such as HTML, Perl, JSP, XML, PHP, etc.

At the lowest level, software consists of a *machine language* specific to an individual processor. A machine language consists of groups of binary values signifying processor instructions that change the state of the computer from its preceding state. Software is an ordered sequence of instructions for changing the state of the computer hardware in a particular sequence. It is usually written in high-level programming languages that are easier and more efficient for humans to use than machine language. High-level languages are compiled or interpreted into machine language object code. Software may also be written in an assembly language that are mnemonic representations of a machine language using a natural language alphabet. Assembly language must be assembled into object code via an assembler.

Software Execution

Once the software is loaded in internal memory or in a drive, the computer is able to *execute* the software. Execution involves passing instructions from the application software, through the system software, to the hardware that ultimately receives the instruction as machine code. Each instruction causes the computer to carry out an operation—moving data, carrying out a computation, or altering the flow of instructions.

Data movement is typically from one place in memory to another. Sometimes, it involves moving data between memory devices and registers, which enables high-speed data access in the CPU. Computations may include simple operations such as incrementing the value of a variable data element.

Software Library

Software library is a collection of executable software but not sufficiently complete for direct execution. Library includes include collections of functions and functionality that may be embedded in other applications. Operating systems include many standard software libraries, and applications are often distributed with their own libraries.

SOFTWARE ARCHITECTURE

Software can be divided into three major groups:

1. System software
2. Programming software
3. Application software

System Software

System software (also called platform software) runs the computer hardware and computer system. It helps the user by avoiding complex details of the operations, including such accessories as communications devices, printers, device readers, displays, utilities, and keyboards. It allows the partitioning of the computer's resources such as memory and processor time in a safe and stable manner. System software includes the operating system, device drivers, and the firmware. Some examples of system software are Windows XP, Linux, and Mac.

Programming Software

Programming software provides tools to assist a programmer in developing operation, and applications programs and software suing a range of languages. Programming software tools include compilers, debuggers, interpreter linkers, and text editors.

Application Software

Application software consists of programs that allow the end users to accomplish one or more specific tasks such as industrial automations, business software, telecommunications, Internet, databases, spreadsheets, word processing, project management, accounting, enterprise computing, PLC programming, SCADA programming, etc. Application software is available in various forms, such as

- *Packaged software* meets the needs of many users, therefore they are mass produced. Examples are computer-aided design, spreadsheets, document management, etc.
- *Custom software* is developed to perform specific tasks of a business or industry.
- *Web application software* allows user to access and interact any computer connected to Internet. Examples are e-mail, word processing, game programs, etc.
- *Open source software* is the software provided for use, modification, and redistribution.
- *Shareware* is copyrighted software that is distributed without cost for trial period. Sometimes scaled-down version of the software is redistributed and payment is required for the full version.
- *Freeware* is copyrighted provided at no cost by an individual or a company.
- *Public domain software* has been donated for public use without copyright restrictions.

Application software is purchased separately from computer hardware. Sometimes applications are bundled with the computer, but that does not change the fact that they run as independent applications. Applications are usually independent programs from the operating system, though they are often tailored for specific platforms.

PROGRAMMING LANGUAGES

A *computer program* is a sequence of instructions for computers to interpret and execute. Computer programs are written by human programmers. Programming is the process of writing, testing, debugging/troubleshooting, and maintaining the source code of computer programs.

A *programming language* is an artificial language for expressing computations that can be performed by microprocessors or computers. Programs are created by using programming languages, some of which will be explained next.

High- and Low-Level Languages

For microprocessors, microcontrollers, and computers, there are a range of languages available. Some of these languages used directly by the process industry or indirectly by the vendors will be explained here. The choice of programming language depends on the application, compactness, and speed of execution, ease of use and maintenance, and availability of compiler or interpreter for the machine in question.

The lowest level of programming is the machine code in binary values as it appears on program memory. In modern applications, the need of machine code is eliminated by assembly programming, which is a set of instructions in mnemonic form. Unlike high-level languages, the assembly language is still difficult to write, to debug, and to understand. Usually, one statement written in a high-level language corresponds to many assembly-level instructions.

Assembly language programming yields to efficiency in execution and compactness as it does not need compilers or interpreters. It needs assembler and debugger to produce functional programs. However, programs developed in assembly languages are not transportable from one microprocessor system to another. Nevertheless, assembly languages are still used in process industry to program dedicated controllers.

A typical program developed in assembly language will look something like this:

```
LOOP 1 LD   (HL),E
            INC      HL
            DEC      BC
            LD       A, B
            OR       C
            JR       NZ, LOOP1
            INC      E
            LD       HL, 0FFFFH
            END
```

BASIC Language

Basic (Beginners All Purpose Symbolic Instruction Code) aims at the development of programs using simple English words rather than abstract symbols. It exists in many forms such as M-BASIC, C-BASIC, X-BASIC, and so on. A typical program written in BASIC will look like this:

```
10    LET n=(VAL(r$)
20    DIM x(n)
30    FOR i=1 TO n
40    INPUT "value"; r$
50    LET x(i)=VAL(r$)
60    NEXT i
70    GOSUB 1000
.     .
.     .
.     .
510   RETURN
```

C Language

C language is highly flexible and portable as it uses a relatively small amount of processor-dependent codes. Portable C compiler (PCC) is popular with Unix, Zilog, Onyx, Xenix, and so on. Lattice C compiler is popular with IBM and similar systems. Two widely accepted forms of C are the C-BASIC, C++, and C#. A sample program written in C is as follows:

```
Main ()
{
  printf ("Press <Q> to quit\n");
  while ((c=getc (stdin)) !=QUIT)
  if (isalnum©)
  decide();
  pause;
  verify();
}
```

Other languages are: COBOL, FORTH, FORTRAN, APL, LOGO, PASCAL, PL/M, and many others.

SOFTWARE STANDARDS, QUALITY, LICENSES, AND PATENTS

Since software can be designed using many different programming languages and in many different operating systems, software standard is needed so that different software can understand and exchange information between each other. There are many organizations like the W3C, IETF, and others that set with standards so that different software can work and interoperate with each other, such as FTP, HTTP, and HTML.

Quality of Software and Reliability

Software quality is very important, especially for commercial applications and system software like Microsoft Office and Linux operating system. Faults and errors are called "bugs." Many bugs can be discovered and eliminated by debugging through software testing. All major software companies, such as Microsoft, Novell, and Sun Microsystems, have their own software testing departments with the specific goal of just testing. Software can be tested through unit testing, regression testing, and other methods. Bugging can be done manually or automatically since the amount of code to be tested can be quite large.

Software Licenses

The software's license gives the user the right to use the software in the licensed environment. Some software comes with the license when purchased off-the-shelf, or an OEM license when bundled with hardware. Other software comes with a free software license, giving the user the rights to modify and redistribute the software. Software can also be in the form of freeware or shareware.

Software Patents

Software can be patented; however, patents can be controversial in the software industry with many people holding different views about it. The controversy over software patents is that a specific algorithm or technique that the software has and may not be duplicated by others. It is considered an intellectual property and copyright. Software patents may be thought of as hindering the development or providing incentives for innovation.

SOFTWARE DESIGN AND IMPLEMENTATION

Design and implementation of software varies depending on the complexity. For example, the design and creation of word processor will take much longer time than design and development of notepad because of the complexities in their functionalities.

Software is usually designed and created in integrated development environment (IDE) such as Eclipse, Emacs, and Visual Studio. IDE simplifies the process and compiles the program. Software is usually created on top of existing software and the application programming interface (API). The underlying software and libraries (APIs) are categorized for different purposes. For example, JavaBeans library is used for designing enterprise applications, Window Forms library is used for designing GUI applications, and Windows Communication Foundation libraries are used for designing web services.

SOFTWARE DEVELOPMENT TOOLS

A *programming tool* or *software development tool* is a program that software developers use to create, debug, maintain, or support other programs and applications. The term refers to relatively simple programs that can be combined together to accomplish a task. Software tools include performance analysis tools, debugging tools, verification tools, checking tools, memory usage tools, and application build tools.

Bibliography

Cherry, E., *Programming for Design: From Theory to Practice*, New York: Wiley, 1998.

Frank, T. and Orlando, K., *Essentials of Software Engineering*, Sudbury, MA: Jones and Bartlett Publishers, 2007.

Pfleeger, S.L. and Atlee, J.M., *Software Engineering: Theory and Practice*, Upper Saddle River, NJ: Pearson–Prentice Hall, 2010.

Pratt, T.W. and Zelkowitz, M.V., *Programming Languages: Design and Implementation*, Upper Saddle River, NJ: Prentice Hall, 1996.

Sethi, R., *Programming Languages: Concepts and Constructs*, Reading, MA: Addison Wesley, 1996.

Sommerville, I., *Software Engineering*, Harlow, U.K.: Addison Wesley, 2007.